48장. 모듈

48.1 모듈의 일반적 의미

1. **모듈**이란 애플리케이션을 구성하는 개별적 요소로서 재사용 가능한 코드 조각을 말한다. 일반적으로 모듈은 기능을 기준으로 파일 단위로 분리한다. 모듈은 자신만의 **파일 스코프(모듈 스코프)**를 가질 수 있어야 한다.

2. 파일 스코프를 갖는 모듈의 모든 자산은 캡슐화 되어 다른 모듈에서 접근할 수 없다. 모듈은 애플리케이션이나 다른 모듈에 의해 재사용 되어야 의미가 있다. 모듈은 공개가 필요한 자산에 한정하여 명시적으로 선택적 공개가 가능하다. 이를 **export**라 한다.

3. 모듈 사용자는 모듈이 공개(export)한 자산 중 일부 또는 전체를 서낵해 자신의 스코프 내로 불러들여 재사용할 수 있다. 이를 **import**라 한다.

4. 모듈은 기능별로 분리되어 개별적인 파일로 작성된다. 따라서 **재사용성이 좋아서 개발 효율성과 유지보수성을 높일 수 있다.**

48.2 자바스크립트와 모듈

처음 자바스크립트는 모듈 시스템(export, import)를 지원하지 않았다. 자바스크립트 파일을 여러 개의 파일로 분리하야 script 태그로 로드해도 분리된 자바스크립트 파일들은 결국 하나의 자바스크립트 파일 내에 있는 것처럼 동작한다.

이런 상황에 제안된 것이 **CommonJS**와 AMD다. 자바스크립트 런타임 환경인 Node.js는 모듈 시스템의 사실상 표준인 CommonJS를 선택했다. 즉, Node.js는 ECMAScript 표준 사양은 아니지만 모듈 시스템을 지원한다.

48.3 ES6 모듈(ESM)

IE를 제외한 대부분의 브라우저에서 ES6 모듈을 사용할 수 있다.

ESM의 사용법

script 태그에 type="module" 어트리뷰트를 추가하면 로드된 자바스크립트 파일은 모듈로서 종작한다.

```
<script type="module" src="app.mjs"></script>
```

48.3.1 모듈 스코프

ESM은 독자적인 모듈 스코프를 갖는다. mjs 파일 내에서 var키워드로 선언한 변수는 더는 전역 변수가 아니며 window 객체의 프로퍼티도 아니다.

```
// foo.mjs
const x = 'foo'
console.log(x) // foo
```

```
// bar.mjs
console.log(x) // ReferenceError: x is not defined
```

48.3.2 export 키워드

- 모듈 내부에서 선언한 식별자를 외부에 공개하며 다른 모듈들이 재사용할 수 있게 하려면 export 키워드를 사용한다.
- 변수, 함수, 클래스 등 모든 식별자를 export 할 수 있다.

48.3.3 import 키워드

- 다른 모듈에서 export한 식별자를 자신의 모듈 스코프 내부로 로드하기 위해 import 키워드 사용
- ESM의 경우는 파일 확장자 생략할 수 없다.
- 모듈에서 하나의 값만 export 하면 default 키워드를 사용할 수 있다. default 키워드를 사용하는 경우 기본적으로 이름 없이 하나의 값을 export 한다.
  - `default` 키워드 사용 시 `var, let, const` 키워드는 사용할 수 없다.
  - default 키워드와 함께 export 한 모듈은 { } 없이 임의의 이름으로 import한다.

```
// app.mjs
// lib.mjs 모듈이 export한 모든 식별자를 lib 객체의 프로퍼티로 모아 import한다.
import * as lib from './lib.mjs'

console.log(lib.pi) // 3.141592...
```